Revisiting Wedge Sampling for Budgeted Maximum Inner Product Search (Extended Abstract)\footnote{This is an extended abstract of work published as [Lorenzen and Pham, 2020], that won the best data mining paper award at the 2020 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases.}

Stephan S. Lorenzen\textsuperscript{1}, Ninh Pham\textsuperscript{2}

\textsuperscript{1}Department of Computer Science, University of Copenhagen
\textsuperscript{2}School of Computer Science, University of Auckland
lorenzen@di.ku.dk, ninh.pham@auckland.ac.nz

Abstract
Top-$k$ maximum inner product search (MIPS) is a central task in many machine learning applications. This work extends top-$k$ MIPS with a budgeted setting, that asks for the best approximate top-$k$ MIPS given a limited budget of computational operations. We study recent advanced sampling methods, including wedge and diamond sampling, to solve budgeted top-$k$ MIPS. First, we theoretically show that diamond sampling is essentially a combination of wedge sampling and basic sampling for top-$k$ MIPS. Second, we propose dWedge, a simple deterministic variant of wedge sampling for budgeted top-$k$ MIPS. Empirically, dWedge provides significantly higher accuracy than other budgeted top-$k$ MIPS solvers while maintaining a similar speedup.

1 Introduction
Maximum inner product search (MIPS) is the task of, given a point set $X \subset \mathbb{R}^d$ of size $n$ and a query point $q \in \mathbb{R}^d$, finding the point $p \in X$ such that,

$$p = \arg \max_{x \in X} x^\top q.$$  

MIPS and its variant top-$k$ MIPS, which finds the top-$k$ largest inner product points with a query, are central tasks in the retrieval phase of standard collaborative filtering based recommender systems [Cremonesi et al., 2010; Koren et al., 2009]. They are also algorithmic ingredients in a variety of machine learning tasks, for instance, prediction tasks on multi-class learning [Dean et al., 2013; Russakovsky et al., 2015] and neural network [Covington et al., 2016; Spring and Shrivastava, 2017].

Modern real-world online recommender systems often deal with very large-scale data sets and a limited amount of response time. Such collaborative filtering based systems often present users and items as low-dimensional vectors. A large inner product between these vectors indicates that the items are relevant to the user preferences. The recommendation is often performed in the online manner since the user vector is updated online with ad-hoc contextual information only available during the interaction [Bachrach et al., 2014; Koenigstein et al., 2011]. A personalized recommender needs to infer user preferences based on the online user behavior, e.g. recent search queries and browsing history, as implicit feedback to return relevant results [Hu et al., 2008; Rendle et al., 2009]. Since the retrieval of recommended items is only performed online, the result of this task might not be "perfect" given a small amount of waiting time but its accuracy should be improved given more waiting time. Hence, it is challenging to not only speed up the MIPS process, but to trade the search efficiency for the search quality.

Motivated by the computational bottleneck in the retrieval phase of modern recommendation systems, we study the budgeted MIPS problem, a natural extension of MIPS with an explicit computational limit for the search efficiency and quality trade-off. Our budgeted MIPS addresses the following question:

Given a data structure built in $\tilde{O}(dn)$ time\footnote{Polylogarithmic factors, e.g. $\log d \log n$ is absorbed in the $\tilde{O}$-notation.} and budgeted computational operations, can we have an algorithm to return the best approximate top-$k$ MIPS?

To measure the accuracy of approximate top-$k$ MIPS, we use the search recall, i.e. the empirical probability of retrieving the true top-$k$ MIPS. In our budgeted setting, we limit the time complexity of building a data structure to $\tilde{O}(dn)$ since when a context is used in a recommender system, the learning phase cannot be done entirely offline [Bachrach et al., 2014; Koenigstein et al., 2011]. In other words, the item vectors are also computed online and hence a high cost of constructing the data structure will significantly degrade the performance.

It is worth noting that the budgeted MIPS has been recently studied in [Yu et al., 2017] given a budget of $B = \eta n$ inner product computation where $\eta$ is a small constant, e.g. 5\%. Furthermore, such budget constraints on the number of computational operations or on accessing a limited number of data points are widely studied not only on search problems [Ram et al., 2012] but also on clustering [Mai et al., 2013; Shamir and Tishby, 2011] and other tasks [Fetaya et al., 2015;
2 Sampling Methods for Top-\(k\) MIPS

For notation, we present the point set \(X\) as a matrix \(X \in \mathbb{R}^{n \times d}\) where each point \(x_i\) corresponds to the \(i\)th row, and the query point \(q\) as a column vector \(q = (q_1, \ldots, q_d)^T\). We use \(i \in [n]\) to index row vectors of \(X\), i.e. \(x_i = (x_{i1}, \ldots, x_{id}) \in \mathbb{R}^d\). Since we will describe sampling methods using the column-wise matrix-vector multiplication \(Xq\), we use \(j \in [d]\) to index column vectors of \(X\), i.e. \(y_j = (x_{1j}, \ldots, x_{nj})^T \in \mathbb{R}^n\). For each column \(j\), we pre-compute its 1-norm \(c_j = \|y_j\|_1\).

We briefly review sampling approaches for estimating inner product contributions are as follows:

\begin{equation}
Xq = \begin{bmatrix} x_{11} \\ \vdots \\ x_{n1} \end{bmatrix} q_1 + \begin{bmatrix} x_{12} \\ \vdots \\ x_{n2} \end{bmatrix} q_2 + \cdots + \begin{bmatrix} x_{1d} \\ \vdots \\ x_{nd} \end{bmatrix} q_d \\
= Y_1q_1 + Y_2q_2 + \cdots + Y_dq_d
\end{equation}

2.1 Basic Sampling

Basic sampling is a very straightforward method to estimate the inner product \(x_i^Tq\) for the point \(x_i\). For any row \(i\), we sample a column \(j\) with probability \(q_j/\|q\|_1\) and return \(x_{ij}\). Define a random variable \(Z_i = x_{ij}\) we have

\[
E[Z_i] = \sum_{j=1}^d x_{ij} q_j/\|q\|_1 = \frac{x_i^Tq}{\|q\|_1}.
\]

The basic sampling suffers large variance when most of the contribution of \(x_i^Tq\) are from a few coordinates. In particular, the variance will be significantly large when the main contributions of \(x_i^Tq\) are from a few coordinates \(x_{ij}q_j\) and \(q_j\) are very small. Note that this basic sampling approach has been used in [Drineas et al., 2006] as an efficient sampling technique for approximating matrix-matrix multiplication.

2.2 Wedge Sampling

[Cohen and Lewis, 1999] proposed an efficient sampling approach, called wedge sampling, to approximate matrix multiplication and to isolate the largest inner products as a byproduct. Wedge sampling needs to pre-compute some statistics, including the sum of all inner products \(z = \sum_i z_i\) where \(z_i = x_i^Tq\) and 1-norm of column vectors \(c_j = \|y_j\|_1\). Since we can pre-compute \(c_j\) before querying, computing \(z = \sum_j q_jc_j\) clearly takes \(O(d)\) query time. We can think of \(q_jc_j/z\) as the contribution ratio of the column \(j\) to the sum of inner product values \(z\).

The basis idea of wedge sampling is to randomly sample a row index \(i\) corresponding to \(x_i\) with probability \(z_i/z\). Hence, the larger the inner product \(z_i = x_i^Tq\), the larger the number of occurrences of \(i\) in the sample set. Consider Equation (1), wedge sampling first samples a column \(j\) corresponding to \(y_j\) with probability \(q_jc_j/z\), and then samples a row \(i\) corresponding to \(x_i\) from \(y_j\) with probability \(x_{ij}/c_j\). By Bayes’ theorem, we have

\[
Pr[\text{Sampling } i] = \sum_{j=1}^d Pr[\text{Sampling } i|\text{Sampling } j] \cdot Pr[\text{Sampling } j] = \sum_{j=1}^d \frac{x_{ij}q_j}{c_j} = \frac{\sum_{j=1}^d x_{ij}q_j}{z} = \frac{z_i}{z}
\]

Applying wedge sampling on \(Xq\), we obtain a sample set where each index \(i\) corresponding to \(x_i\) is sampled according to an independent Bernoulli distribution with parameter \(p_i = z_i/z\). To answer top-\(k\) MIPS, we execute wedge sampling and find the points with the largest counters. Given \(S\) samples and a constant cost for each sample, wedge sampling runs in \(O(S + \min(S,n)\log k)\) time to answer approximate top-\(k\) MIPS. If we have an additional budget of \(B > k\) inner product computation, we can compute the inner product values of the top-\(B\) points with the largest counter values for ranking. Such ranking (or post-processing) phase with an additional \(O(dB)\) cost will provide higher accuracy for top-\(k\) MIPS.

We note that since wedge sampling uses the contribution ratio \(q_jc_j/z\) to sample the column \(j\), it can alleviate the effect of skewness of \(x_i^Tq\) where large contributions are from a few coordinates. Hence wedge sampling achieves lower variance than the basic sampling in practice.

2.3 Diamond Sampling

[Ballard et al., 2015] proposed diamond sampling to find the largest magnitude elements from a matrix-matrix multiplication \(XQ\) without computing the final matrix directly. The method considers \(XQ\) as a weighted tripartite graph, and samples a diamond, i.e. four cycles from this graph with probability proportional to the value \((XQ)_{ij}\) in order to amplify the focus on the largest magnitude elements.

Consider a vector \(q\) as a one-column matrix \(Q\), it is clear that diamond sampling can be applied to solve MIPS. Indeed, we will show that diamond sampling is essentially a combination of wedge sampling and basic sampling when approximating \(XQ\). In particular, diamond sampling first makes use
of wedge sampling to return a random row \( i \) corresponding to \( X_i \) with probability \( z_i / z \). Given such row \( i \), it then applies basic sampling to sample a random column \( j \) with probability \( q_i / \|q\|_1 \) and return \( x_{ij} \) as a scaled estimate of \( (x_i^T q)^2 \). Define a random variable \( Z_i = x_{ij} \) corresponding to \( X_i \), using the properties of wedge sampling and basic sampling we have

\[
E[Z_i] = \sum_{j=1}^d x_{ij} q_j / \|q\|_1 \cdot z_i = \frac{(x_i^T q)^2}{z \|q\|_1}.
\]

Since diamond sampling builds on basic sampling, it suffers from the same drawback as basic sampling. To answer top-\( k \) MIPS, diamond follows the same procedure as wedge hence shares the same asymptotic running time.

### 3 Wedge Sampling for Budgeted Top-\( k \) MIPS

We first show a concentration bound of wedge sampling, which shows that wedge requires fewer samples than diamond for approximating top-\( k \) MIPS. Then we present a drawback of wedge sampling for the budgeted MIPS and propose \( dWedge \), a simple deterministic variant to handle this drawback. \( dWedge \) can govern the trade-off between search quality and efficiency with two parameters: the number of samples \( S \) and the number of inner product computations \( B \).

#### 3.1 A New Analysis of Wedge Sampling

This subsection presents the analysis of wedge sampling on non-negative inputs. Consider a counting histogram of \( n \) counters corresponding to \( n \) point indexes, the following theorem states the number of samples required to distinguish between two inner product values \( \tau_1 \) and \( \tau_2 \).

**Theorem 1.** Fix two thresholds \( \tau_1 > \tau_2 > 0 \) and suppose \( S \geq \frac{3z \ln n}{\sqrt{\tau_1 - \sqrt{\tau_2}}} \) where \( z = \sum_i x_i^T q \). With probability at least \( 1 - \frac{1}{n} \), the following holds for all pairs \( i_1, i_2 \in [n] \): if \( x_{i_1}^T q \geq \tau_1 \) and \( x_{i_2}^T q \leq \tau_2 \), then \( \text{counter}[i_1] > \text{counter}[i_2] \).

The detailed proof can be found in [Lorenzen and Pham, 2020]. We will discuss some implications of Theorem 1.

**Trade-off between search quality and efficiency.** By choosing \( S \) as in Theorem 1, we have \( \sqrt{\tau_1} - \sqrt{\tau_2} \geq \sqrt{3z \ln n} / S \). Assume that the top-\( k \) value is \( \tau_1 \), it is clear the more samples we use, the smaller gap between the top-\( k \) MIPS values and the other values we can distinguish. We note that we can compute and rank \( B \) inner products of the top-\( B \) points with the largest counter values in the ranking phase. Increasing \( B \) corresponds to increasing the gap \( \tau_1 - \tau_2 \). This means that both \( B \) and \( S \) can be used to control such trade-off.

**Comparison to diamond sampling.** For a fair theoretical comparison, we consider the same setting as in [Ballard et al., 2015, Theorem 4] where we want to distinguish \( x_{i_1}^T q \geq \tau \) and \( x_{i_2}^T q \leq \tau /4 \), and all entries in \( X \) and \( q \) are non-negative. 2 Applying Theorem 1, diamond sampling needs \( S_w = 12 \ln n / \tau \). Diamond sampling needs \( S_d = 12K / \|q\|_1 \ln n / \tau^2 \) where all entries in \( X \) are at most \( K \). Since \( K / \|q\|_1 \geq \tau \) for any \( \tau \), wedge requires strictly less samples than diamond.

---

#### 3.2 dWedge: A Simple Deterministic Variant for Budgeted MIPS

This subsection presents a significant drawback of wedge sampling for solving top-\( k \) MIPS with a \( o(n) \) budgeted computation. We then introduce \( dWedge \), a simple deterministic variant to handle this drawback. For simplicity, we present our approach on non-negative \( X \) and \( q \).

**Drawback.** We observe that wedge sampling first samples the column \( j \) and then samples the point \( X_i \) for this column \( j \). In other words, given a fixed number of samples \( S \), wedge sampling allocates \( S \) samples to \( d \) columns. Each column \( j \) receives \( s_j \) samples and \( \sum_j s_j = S \). To ensure that wedge provides an accurate estimate, these \( s_j \) samples must approximate the discrete distribution \( y_j / \|y_j\|_1 \) well. Given the budget of \( S = o(n) \) samples, the number of samples \( s_j \) of the column \( j \) is roughly \( S / d \ll n \) in expectation. Since \( s_j \ll n \) and the data set is often dense, it is impossible to approximate the discrete distribution \( y_j / \|y_j\|_1 \) by \( s_j \) samples. Hence the performance of wedge (and hence diamond) sampling dramatically degrades in the budgeted setting.

**dWedge.** Observing that wedge sampling carefully distributes \( S \) samples to each dimension. Dimension \( j \) receives \( s_j = Sc_j q_j / z \) samples and hence the point \( X_i \) for dimension \( j \) will receive \( s_j x_{ij} / c_j \) samples in expectation. Given \( S = o(n) \), we have \( s_j \ll n \) and therefore can only sample a few points for dimension \( j \). Due to this limit, we propose to greedily sample \( X_i \) with the largest \( x_{ij} \) values in column \( j \). For each selected \( X_i \), we sample \( s_j x_{ij} / c_j \) times. Algorithm 1 presents our simple heuristic solution for non-negative \( X \) and \( q \). For negative inputs, \( dWedge \) executes on the absolute values of \( X \) and \( q \). If selected, the counter for \( X_i \) is increased by \( sgn(x_{ij}) sgn(q_j) [s_j x_{ij} / c_j] \).

**Time complexity.** It is clear that the pre-processing step takes \( O(dn \ln n) \) time and \( O(dn) \) additional space. \( dWedge \) sampling takes \( O(S) \) time. If \( S \ll n \), we can use a hash table to maintain the counting histogram. Otherwise, we use a vector of size \( n \). The running time of \( dWedge \) for answering top-\( k \) MIPS with a post-processing \( B \) inner products is \( O(S + min(S, n) \log B + dB) \).

**Relation to Greedy-MIPS.** Greedy-MIPS [Yu et al., 2017] exploits the upper bound \( x_i \cdot q \leq d \max_j \{q_j x_{ij}\} \) to construct...
B candidates. In principle, it greedily selects B points \(x_i\) with the largest \(q_j x_{ij}\) values for each dimension \(j\), then merges them to find top-\(B\) candidates with \(\max_j \{q_j x_{ij}\}\). While dWedge shares the same pre-processing step and greedy spirit, there is a significant difference between dWedge and Greedy. dWedge greedily estimates and differentiates the largest elements on each dimension \(j\) using \(s_j\) samples. The more samples used, the more largest elements considered. This leads to a higher quality of top-\(B\) candidates and top-\(k\) MIPS.

4 Experiment

Since diamond exploits wedge, applying dWedge to diamond derives a new variant, called dDiamond. Our implemented algorithms include: (1) The traditional wedge (Wedge) and diamond (Diamond) sampling and the proposed solutions dWedge and dDiamond; (2) The Greedy-MIPS approach (Greedy) [Yu et al., 2017]; and (3) Brute-force algorithm with the Eigen-3.3.4 library for the fast C++ matrix-vector multiplication.

We conduct experiments on standard real-world data sets, including Netflix \((n = 17,770; d = 300)\) and Yahoo \((n = 624,961; d = 300)\) from [Cremonesi et al., 2010], and Gist \((n = 1,000,000; d = 960)\). For Netflix and Yahoo, the item matrices are used as the data sets. We randomly pick 1000 users from the user matrices to form the query sets. All randomized results are the average of 5 runs of the algorithms.

Wedge vs. Diamond. Figure 2 reveals that the proposed deterministic generator gives higher accuracy than the randomized one. dWedge and dDiamond outperform Wedge and Diamond, respectively, over a wide range of \(S\). In term of speedup, dWedge runs significantly faster than dDiamond, especially at least twice faster when \(S = n\). Wedge variants run faster than diamond variants since diamond builds on basic sampling that requires expensive cost for random accesses.

dWedge vs. Greedy-MIPS. Note that dWedge’s cost is about \(2S/d + B\) inner product computations. Since we want to show that dWedge always runs as fast as Greedy but achieves higher accuracy, we use \(2S/d + B\) inner product computations for Greedy. Given this setting, Figure 1 shows that dWedge runs faster and provides dramatically higher accuracy than Greedy on Yahoo and Gist. On Yahoo, since the screening phase of Greedy incurs remarkable cost, dWedge yields substantially higher accuracy and speedup. In order to show the benefit of the sampling phase of dWedge, we measure its performance on Gist while fixing \(B = 200\) and varying \(S\). Since we add additional inner product computation into the post-processing phase, dWedge and Greedy achieve similar speedup, as shown in Figure 1 (c). Figure 1 (d) shows that Greedy suffers from very small accuracy, achieving at most 40%, while dWedge achieves nearly perfect recall, i.e. 99% when \(S = 2n\). In general, dWedge with two parameters, i.e. number of samples \(S\) and number of inner products \(B\), governs the trade-off between search efficiency and quality more efficiently than Greedy.

dWedge vs. Locality-sensitive Hashing. Detailed comparisons between dWedge and recent advanced locality-sensitive hashing methods for budgeted top-\(k\) MIPS [Neyshabur and Srebro, 2015; Yan et al., 2018] can be found in [Lorenzen and Pham, 2020].

5 Conclusion

This paper studies top-\(k\) MIPS given a limited budget of computational operations and investigates recent advanced sampling algorithms, including wedge and diamond sampling, for solving it. We theoretically and empirically show that wedge sampling is competitive (often superior) to diamond sampling for approximating top-\(k\) MIPS regarding both efficiency and accuracy. We propose dWedge, a simple deterministic wedge-based variant for the budgeted top-\(k\) MIPS. Empirically, dWedge runs significantly faster than other competitive budgeted MIPS solvers on real-world benchmark data sets while maintaining a very competitive accuracy.
References


